# Deliverable D3.4

# UNITE Package


| | |
|---|---|
| CONTRACT NO | HOPSA-EU 277463 |
| INSTRUMENT | CP (Collaborative project) |
| CALL | FP7-ICT-2011-EU-Russia |

Due date of deliverable: December 1st, 2012

Actual submission date: April 5th, 2012

Start date of project: 1 FEBRUARY 2011 Duration: 24 months

Name of lead contractor for this deliverable: JUELICH


**Abstract**: The goal of **UNITE** (**UN**iform **I**ntegrated **T**ool **E**nvironment) is to provide a robust, portable, and integrated environment for the debugging and performance analysis of parallel MPI, OpenMP, and hybrid MPI/OpenMP programs on high-performance compute clusters. It consists of a set of well-accepted portable, mostly open-source tools, together with a "meta"-installation tool allowing downloading, configuring, building, and installing all tools as one portable and coherent package.

| **Project co-funded by the European Commission within the Seventh Framework Programme (FP7/2007-2013)** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | X |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

# Table of Contents



  

# Glossary

| Abbreviation / acronym | Description |
|---|---|
| API | Application Programming Interface |
| BSC | Barcelona Supercomputing Center, Spain |
| CEPBA | European Center for Parallelism of Barcelona (UPC, BSC) |
| ClustrX | Cluster monitoring system (T-Platform) |
| CUBE | Performance report explorer for Scalasca (JSC) and Score-P (GRS, JSC, TUD) |
| CUDA | Compute Unified Device Architecture<br>(Proprietary Programming Interface for Nvidia GPGPUs) |
| Dimemas | Message passing performance analysis and prediction tool (BSC) |
| Extrae | Instrumentation and measurement component for Paraver visualizer (BSC) |
| GRS | German Research School for Simulation Sciences GmbH, Aachen, Germany |
| GPGPU | General Purpose Graphical Processing Unit |
| GUI | Graphical User Interface |
| HMPP | Hybrid Multicore Parallel Programming<br>(Proprietary Programming Model for Heterogeneous Architectures) |
| HOPSA | HOlistic Performance System Analysis. EU FP7 project |
| HPC | High Performance Computing |
| H4H | Hybrid Programming For Heterogeneous Architectures. EU ITEA2 project |
| I/O | Input/Output |
| JSC | Jülich Supercomputing Centre<br>(of Forschungszentrum Jülich GmbH), Germany |
| LAPTA | Database and analysis system for cluster monitoring data (MSU) |
| LWM2 | Light Weight Monitoring Module (GRS)<br>(Used for system-wide application performance screening) |
| MPI | Message Passing Interface<br>(Programming Model for Distributed Memory Systems) |
| MSU | Moscow State University |
| OpenCL | Open Computing Language<br>(Programming interface for heterogeneous platforms consisting of CPUs and other execution units like GPUs) |
| OpenMP | Open Multi-Processing<br>(Programming Model for Shared Memory Systems) |
| OTF2 | Open Trace Format Version 2 |

| | |
|---|---|
| PAPI | Performance Application Programming Interface<br>(Library for portable access to hardware performance counter) |
| Paraver | Event trace analysis and visualization tool (BSC) |
| PMPI | Standard monitoring API for MPI |
| RW | Rogue Wave Software AB, Sollentuna, Sweden |
| Scalasca | SCalable Analysis of LArge SCale Applications<br>(Performance instrumentation, measurement and analysis tool from JSC/GRS) |
| Score-P | Scalable Performance Measurement Infrastructure for Parallel Codes<br>(Community open-source project of GRS, JSC, TUD and others) |
| SMPSs | Pragma-based programming model for parallel task (Ss = Superscalar)<br>for shared memory parallel computers (SMP) from BSC |
| UPC | Universitat Politècnica de Catalunya, Barcelona |
| T-Platforms | Russian HPC cluster vendor |
| ThreadSpotter | Commercial memory and multi-threading performance analysis tool (RW) |
| TUD | Technische Universität Dresden, Germany |
| UNITE | UNiform Integrated Tool Environment<br>(Unified documentation and installation procedures for HPC tools) |
| Vampir | Visualization and Analysis of MPI Resources<br>(Commercial event trace analysis and visualization tool from ZIH/TUD) |
| VampirTrace | Instrumentation and measurement component for Vampir visualizer (ZIH/TUD) |
| ZIH | Zentrum für Informationsdienste und Hochleistungsrechnen.<br>(Center for information services and HPC of TUD). |

# 1. Executive summary

The goal of **UNITE** (**UN**iform **I**ntegrated **T**ool **E**nvironment) is to provide a robust, portable, and integrated environment for the debugging and performance analysis of parallel MPI, OpenMP, and hybrid MPI/OpenMP programs on high-performance compute clusters. It consists of a set of well-accepted portable, mostly open-source tools.

High-performance clusters often provide multiple MPI libraries and compiler suites for parallel programming. This means that parallel programming tools which often depend on a specific MPI library, and sometimes on a specific compiler, need to be installed multiple times, once for each combination of MPI library and compiler which has to be supported. In addition, over time, newer versions of the tools get released and installed. One way to manage many different versions of software packages, used by many computing centres all over the world, is the "module" software (see [13]). However, each centre provides a different set of tools, has a different policy on how and where to install different software packages, and how to name the different versions. This makes it harder for **HPC application developers** to use these tools to debug, analyze and optimize the performance of their scientific applications because when they start using a cluster at a new HPC centre, they first need to find out about these local conventions and special instructions on how to use the debugging and performance tools even if they have used the same tools at other sites before.

UNITE tries to improve this situation for debugging and performance tools by

- specifying exactly how and where to install the different versions of tool software packages (including integrating the tools to the maximum possible degree),
- defining standard module names for tools and their different versions, and
- supplying pre-defined module files which provide standardized, well-tested tool configurations,
- **but** still being flexible enough to be able to co-exist with site-local installations, restrictions, and policies.

This way, the HPC application developers only need to learn once how to invoke and facilitate the debugging and performance tools for their applications (at least at sites which use the UNITE package).

**HPC cluster system administrators** have another problem. In order to provide all these tools for their users, for each tool they have to find out where and how to download the tool, and how to configure, build and install it. As often with open-source software, these steps are very different for the different tools. As the administrators are typically not tool users or even tool experts, they do not know much about the specific features provided by the tools, and especially they miss opportunities to install the tools in a way so they are as much as possible integrated with each other. Furthermore, new versions of the tools get released with new features, bug fixes or changes compared to older versions, often as much as twice a year (typically for the ISC conference in June and the SC conference in November), so the tool installations have to be updated all the time (as users of course want to use the latest features).

UNITE tries to help the system administrators by providing a ”meta”-installation tool (the UNITE installer) which is capable of configuring, building, and installing all supported tools as a common package according to the UNITE specifications but hiding tool-specific aspects of the various phases. It can also be used to update an existing UNITE installation with new tools or tool versions.

On the UNITE website [14], we provide a single package consisting of the UNITE installer, the UNITE module files, as well as a large set of open-source debugging and performance tools together with a user guide (for application developers) and an installation guide (for system administrators).

  

This work is based on a successful prototype developed as part of the EU ITEA-2 project ParMA [15] which could handle Scalasca, Cube3, Vampir, and VampirTrace as well as a few other open-source tools (PDToolkit, Marmot). As part of the EU FP7 project HOPSA, this prototype was enhanced in many ways:

- The UNITE module file package was re-designed and re-implemented so that an existing UNITE installation can be updated with newer versions without losing site-specific adaptations and changes.
- Support for configuring, building and installing new packages has been added:
    - The HOPSA light-weight monitoring module LWM$^2$
    - The BSC tools Extrae and Paraver which made it necessary to also support the additional development packages libdwarf, libunwind, boost, wxpropgrid
    - The Score-P components scorep, otf2, cube4, and opari2 as well as scalasca2
    - The TAU package of the University of Oregon (a contributor and member of the Score-P community project)
    - The new binary installation packages of Vampir and VampirServer introduced with version 7
- The much larger number of packages and the trend to “componentize” tool modules allowed to greatly enhancing the integration among the different packages. This required implementing much more complicated configuration, build and install procedures so that for example when installing an updated version of a tools package it is still integrated with already existing and installed versions of other tool components.
- Support for detecting more MPI library variants (MPICH version 3, BullX MPI, IBM POE MPI library for x86 and x86_64 Linux (intelpoe) and PPC Linux (ibmpoe), and IBM Platform MPI) and configuring the packages accordingly.

Finally, various versions of the UNITE package have been extensively tested on a large variety of HPC Linux clusters all over the world.

  

# 2. Introduction

This section summarizes the performance instrumentation, measurement, analysis, visualization, and modelling tools provided by the HOPSA project partners and describes the performance analysis workflow which defines how to use all these tools in an effective and integrated way.

## 2.1 The broader context: The HOPSA project

To maximize the scientific and commercial output of a high-performance computing system, different stakeholders pursue different strategies. While individual application developers are trying to shorten the time to solution by optimizing their codes, system administrators are tuning the configuration of the overall system to increase its throughput. Yet, the complexity of today's machines with their strong interrelationship between application and system performance demands for an integration of application and system programming.

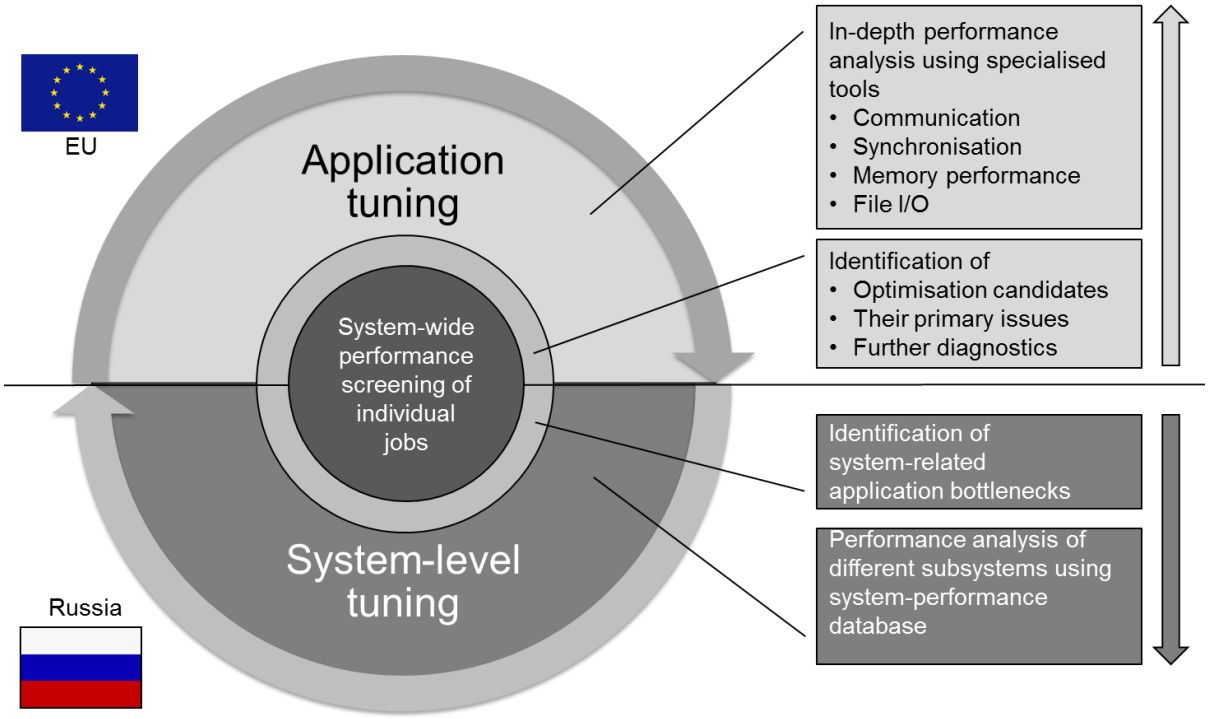


***Figure 1: System-level tuning (bottom), application-level tuning (top), and system-wide performance screening (centre) use common interfaces for exchanging performance properties***

The HOPSA project (HOlistic Performance System Analysis) therefore sets out for the first time for combined application and system tuning in the HPC context developing an integrated diagnostic infrastructure. Using more powerful diagnostic tools, application developers and system administrators can easier identify the root causes of their respective bottlenecks. With the HOPSA infrastructure, it is more effective to optimize codes running on HPC systems. More efficient codes mean either getting results faster or being able to get higher quality or more results in the same time.

The work in HOPSA was carried out by two coordinated projects funded by the EU under call FP7-ICT-2011-EU-Russia and the Russian Ministry of Education and Science. Its objective was the new innovative integration of application tuning with overall system diagnosis and tuning to maximize the scientific output of

  

our HPC infrastructures. While the Russian consortium focused on the system aspect, the EU consortium focused on the application aspect.

At the interface between these two facets of our holistic approach, which is illustrated in Figure 1, will be the system-wide performance screening of individual jobs, pointing at both inefficiencies of individual applications and system-related performance issues.

## 2.2 HOPSA Work package 2: HPC application level analysis

For HPC application tuning, developers can choose from a variety of mature performance-analysis tools developed by the HOPSA-EU consortium: the memory and thread analyzer ThreadSpotter (RW), the trace visualizer Paraver including its measurement system Extrae (BSC), the performance prediction tool Dimemas (BSC), the trace visualizer Vampir (TUD), the performance measurement and analysis tool Scalasca including its result browser Cube (GRS, JSC), and the instrumentation and measurement system Score-P (GRS, JSC, TUD).

### 2.2.1 Short description of the HOPSA performance tools

This section gives a brief overview about the performance instrumentation, measurement and analysis tools within the HOPSA project. For a longer description see the deliverable D2.2 (“Final Tool Set”).

Within work package 2 of the HOPSA project, the tools were further integrated and enhanced with respect to scalability, depth of analysis, and support for asynchronous tasking, a node-level paradigm playing an increasingly important role in hybrid programs on emerging hierarchical and heterogeneous systems. The overall objective of work package 2 was to enhance and extend the already existing individual performance measurement and analysis tools of the project partners to make them fit for the analysis of petascale computations and beyond as well as integrating them with each other where useful. The idea here was not to start new research directions but rather to finalize (i.e., “productize”) current research ideas and make them part of the regular tool products. The tools are available as a combination of open-source offerings (Extrae, Paraver, Dimemas, Scalasca, Cube, Score-P) and commercial products (Vampir, ThreadSpotter).

#### Paraver

Paraver [7][9] is a very flexible data browser that is part of the CEPBA-Tools toolkit developed by BSC and is available for download under an LGPL open-source license. The tool allows a very detailed and powerful exploration of the trace data. Programmable through configuration files, Paraver can visualize performance data via timeline displays (showing metrics per process or thread over time) or histogram displays (showing statistical data). Paraver’s measurement system is Extrae. Extrae is capable of instrumenting applications based on MPI, OpenMP, pthreads, CUDA and StarSs using different instrumentation approaches. The information gathered by Extrae typically includes timestamped events of runtime calls, performance counters and source code references. Besides, Extrae provides its own API to allow the user to manually instrument his or her application.

#### Dimemas

Dimemas [8] is a performance analysis tool for message-passing programs. The Dimemas simulator reconstructs the time behaviour of a parallel application on a machine modelled by the key factors influencing the performance. With a simple model Dimemas allows to simulate complete parametric studies in a very short time frame. Dimemas generates as part of its output a Paraver trace file, enabling the user to conveniently examine the simulator run.

#### Scalasca

Scalasca [1][2][3] supports the performance optimisation of parallel programs by measuring and analysing their runtime behaviour. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronisation – and offers guidance in exploring their causes. The user of Scalasca can choose between two different analysis modes: (i) performance overview on the call-path level via runtime summarisation (aka profiling) and (ii) in-depth study of application behaviour via event tracing. Scalasca, which is jointly developed by JSC and GRS, is available for download under the New BSD open-source license.

  

### Score-P

Score-P [6] is a new open source measurement system developed by TUD and JSC in cooperation with other project partners outside the HOPSA project. Starting 2013, it replaces the measurement systems of Scalasca (EPIK) and Vampir (VampirTrace). It features a new trace (OTF2) and profile (CUBE4) format.

### ThreadSpotter

The ThreadSpotter [4] performance optimization technology is developed by Rogue Wave AB – a spin-out from research at Uppsala University in Sweden. While an ordinary binary is running in a production environment, this new performance debugger collects sparse information about its execution behaviour into a ”fingerprint” file. Based on this information, the cache performance of any size cache, any size cache line and several replacement policies can be estimated off-line. ThreadSpotter’s analysis technology also detects performance bugs in the applications, i.e. certain access patterns that result in a sub-optimal performance. ThreadSpotter organizes such performance bugs into four issue groups: bandwidth issues, latency issues, thread interaction issues and cache pollution issues.

### Vampir

Vampir (”Visualisation and Analysis of MPI Resources”) is a very well-known event trace visualisation software [5] which is available since 1996 as a commercial product. It offers intuitive parallel event trace visualisation with many displays showing different aspects of the parallel performance behaviour. The corresponding VampirTrace instrumentation and run-time measurement package is available as open source. Vampir is developed by ZIH, TU Dresden and is commercially distributed by the technology transfer company GWT-TUD GmbH.

## 2.2.2 HOPSA tool integration

Sharing the common measurement infrastructure Score-P [6] and its data formats, and providing conversion utilities if direct sharing is not possible, the performance tools in the HOPSA environment and workflow already make it easier to switch from higher-level analyses provided by tools like Scalasca to more in-depth analyses provided by tools like Paraver or Vampir. To simplify this transition even further, the HOPSA tools are integrated in various ways (Figure 2). With its automatic trace analysis, Scalasca locates call paths affected by wait states caused by load or communication imbalance. However, to find and fix these problems in a user application, it is in some cases necessary to understand the spatial and temporal context leading to the inefficiency, a step naturally supported by trace visualizers like Paraver or Vampir. To make this step easier, the Scalasca analysis remembers the worst instance for each of the performance problems it recognizes. Then, the Cube result browser can launch a trace browser and zoom the timeline into the interval of the trace that corresponds to the worst instance of the recognized performance problems. In order to allow the use of Paraver for this analysis, the BSC team implemented an OTF2 to Paraver trace format conversion.

In the future, it is planned to use the same mechanisms for a more detailed visual exploration of the results of Scalasca's root cause analysis as well as for further analyzing call paths involving user functions that take too much execution time. For the latter, ThreadSpotter will be available to investigate their memory, cache and multi-threading behaviour. If a ThreadSpotter report is available for the same executable and dataset, Cube will allow launching detailed ThreadSpotter views for each call path where data from both tools is available. The necessary interfaces have been designed and implemented as a prototype during the HOPSA project.

  

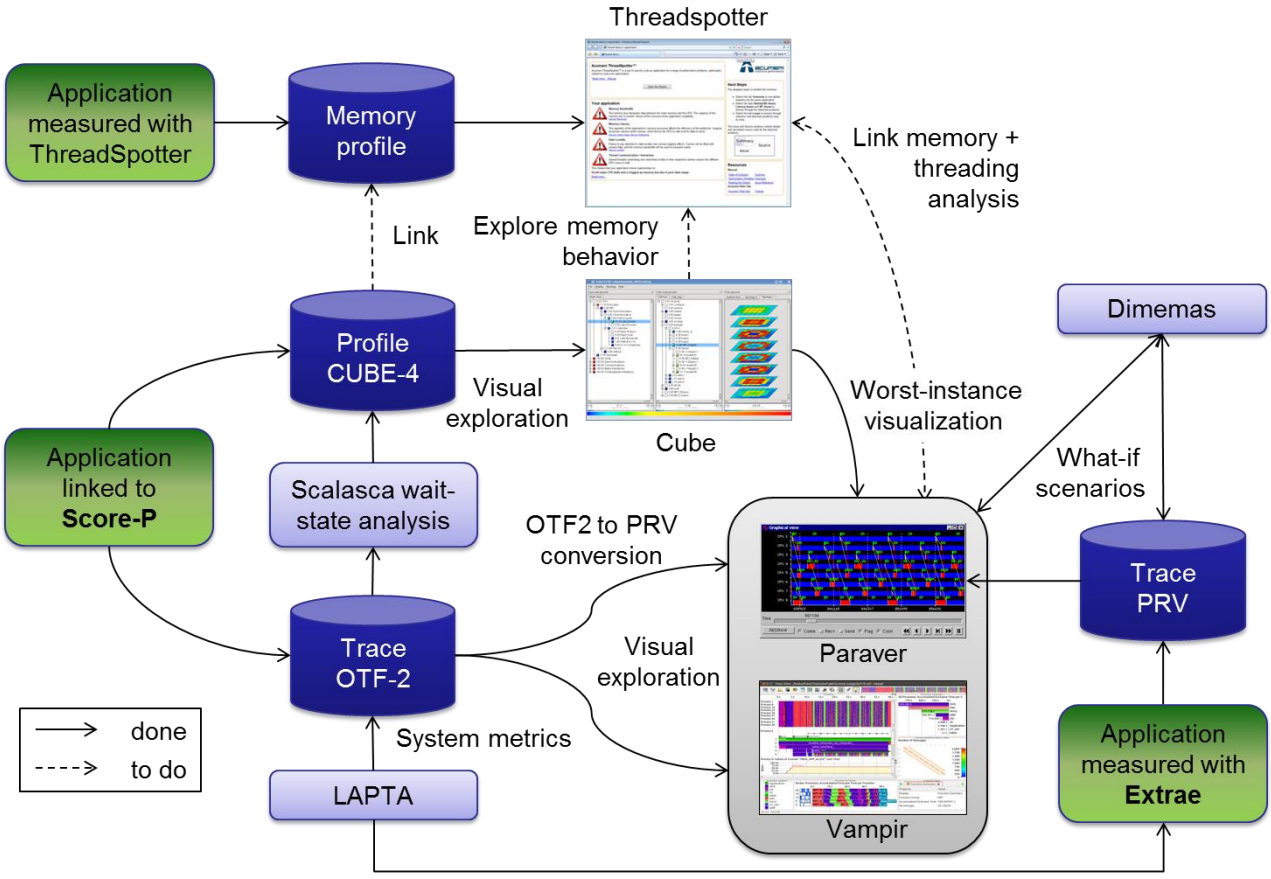


***Figure 2: HOPSA Performance Tool Integration***

Finally, a tight integration of Dimemas and Paraver allows users to investigate various “what-if scenarios” to further analyze performance properties of their applications.

## 2.3 HOPSA Work package 3: Integration of system and application performance analysis

The objective of the HOPSA work package 3 was to combine the work done for the HPC system-level performance analysis (implemented by the Russian partners) and for application-level performance analysis (implemented by work package 2) into a coherent and holistic performance analysis environment. It provides

- Low-overhead and end-to-end performance analysis for all jobs on a given system from their submission to their completion
- Identification of key performance issues and notification of the user and system performance database after job completion
- Detailed scalable performance analysis for petascale applications based on well-accepted and robust measurement and analysis tools.

Key to achieving these objectives was to define an overall performance tool process and workflow which guides **application developers** in the process of tuning and optimizing their codes for performance as well as providing a single package for **system administrators** (or expert users) allowing to configure, built, and install all HOPSA performance tools as one portable and coherent installation.

  

## 2.3.1 The HOPSA workflow

The intended usage and application of the HOPSA performance tools is specified by the HOPSA performance-analysis workflow (Figure 3). It consists of three basic steps. During the first step ("Performance Screening"), we identify all those applications running on the system that may suffer from inefficiencies. This is done via system-wide job screening supported by a lightweight measurement module (LWM$^2$) dynamically linked to every executable. The screening output identifies potential problem areas such as communication, memory, or file I/O, and issues recommendations on which diagnostic tools can be used to explore the issue further in a second step ("Performance Diagnosis"). If a more simple, profile-oriented aggregated performance overview is not enough to pin-point the problem, a more detailed, trace-based, dynamic performance analysis can be performed in a third step ("In-depth analysis") using the HOPSA tool set. The data collected by LWM$^2$ is also fed into the Clustrx.Watch hierarchical cluster monitoring system [10] which combines it with system and hardware data and forwards it to the LAPTA cluster monitoring and analysis system [11] for further analysis by system administrators.

In general, the workflow successively narrows the analysis focus and increases the level of detail at which performance data is collected. At the same time, the measurement configuration is optimised to keep intrusion low and limit the amount of data that needs to be stored. To distinguish between system and application-related performance problems, Paraver and Vampir also allow system-level data to be retrieved and displayed. The system administrator, in contrast, has access to global performance data. He can use this data to identify potential system performance bottlenecks and to optimise the system configuration based on current workload needs. In addition, the administrator can identify applications that consistently underperform and proactively offer performance-consulting services to the effected users. In this way, it facilitates reducing the unnecessary waste of expensive system resources.

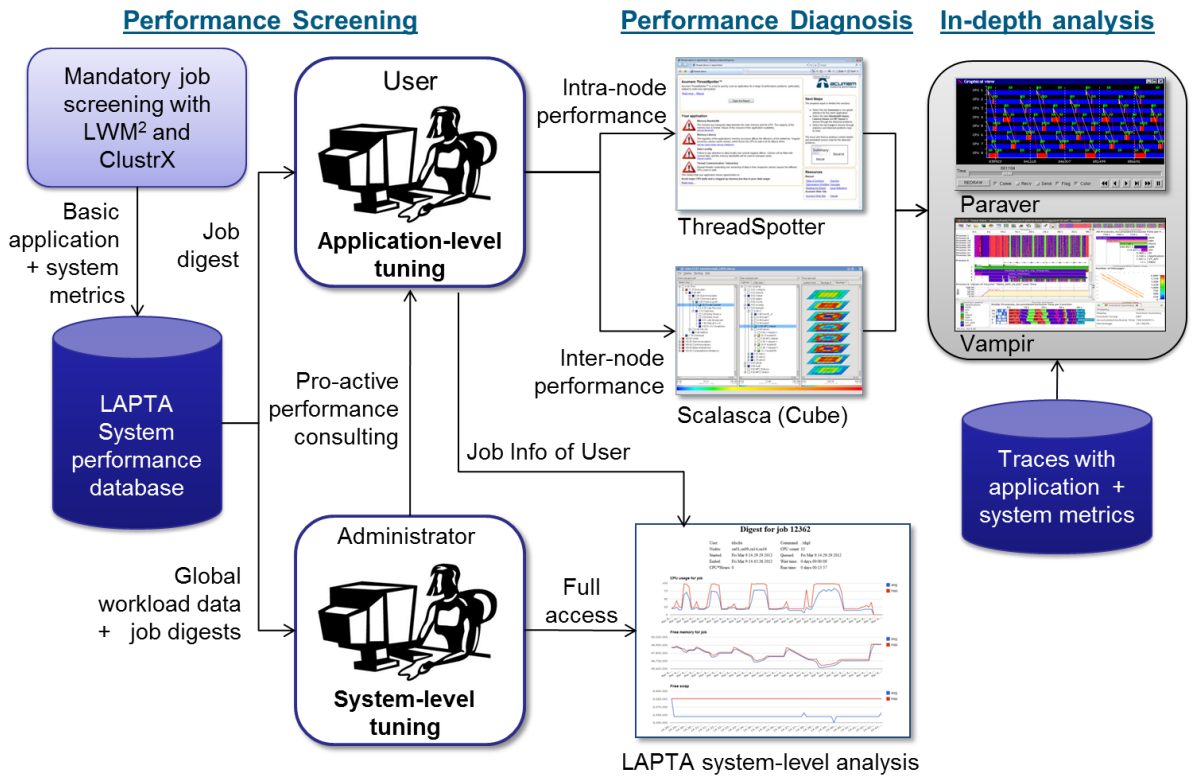


***Figure 3: The HOPSA Performance-Analysis Workflow***

More details about the HOPSA performance-analysis workflow and tool integration can be found in the Deliverable D3.2 ("Workflow Report") as well in [12]. The light-weight measurement module (LWM$^2$) is described in more detail in Deliverable D3.3 ("Light-weight Monitoring Module").

### 2.3.2 The need for a UNITE package

High-performance clusters often provide multiple MPI libraries and compiler suites for parallel programming. This means that parallel programming tools which often depend on a specific MPI library, and sometimes on a specific compiler, need to be installed multiple times, once for each combination of MPI library and compiler which has to be supported. In addition, over time, newer versions of the tools also get released and installed. One way to manage many different versions of software packages, used by many computing centres all over the world, is the "module" software (see [13]). However, each centre provides a different set of tools, has a different policy on how and where to install different software packages, and how to name the different versions. This makes it harder for **HPC application developers** to use these tools to analyze and optimize the performance of their scientific applications because when they start using a cluster at a new HPC centre, they first need to find out about these local conventions and special instructions on how to use the performance tools even if they have used the same tools at other sites before.

UNITE tries to improve this situation for debugging and performance tools by

- specifying exactly how and where to install the different versions of tool software packages (including integrating the tools to the maximum possible degree),
- defining standard module names for tools and their different versions, and
- supplying pre-defined module files which provide standardized, well-tested tool configurations,
- **but** still being flexible enough to be able to co-exist with site-local installations, restrictions, and policies.

This way, the HPC application developers only need to learn once how to invoke and facilitate the performance tools for their applications (at least at sites which use the UNITE package).

**HPC cluster system administrators** have another problem: in order to provide all these tools (for example the HOPSA tool set) for their users, for each tool they have to find out where and how to download the tool, and how to configure, build and install them. As often with open-source software, these steps are often different for the different tools. As they are typically not tool users or even tool experts, they do not know much about the specific features provided by the tools, and especially they miss opportunities to install the tools in a way so they are as much as possible integrated with each other. On top of this, new versions of the tools get released with new features or changes compared to older versions, often as much as twice a year (typically for the ISC conference in June and the SC conference in November), so the tool installations have to been updated all the time (as user of course want to use the latest features).

UNITE tries to help the system administrators by providing a ”meta”-Installation tool (the UNITE installer) which is capable of configuring, building, and installing all supported tools as a common package according to the UNITE specifications but hiding tool-specific aspects of the various phases. It can also be used to update an existing UNITE installation with new tools or tool versions.

On the UNITE website [14], we provide a single package consisting of the UNITE installer, the UNITE module files, as well as a large set of public-domain performance tools together with a user guide (for application developers) and an installation guide (for system administrators).

  

# 3. The UNITE package

The work on a new version (1.1) of UNITE in the HOPSA project was based on a successful prototype (Version 1.0) developed as part of the EU ITEA-2 project ParMA [15] which could handle Scalasca, Cube3, Vampir, and VampirTrace as well as a few other open-source tools (PDToolkit, Marmot). This section provides more details on the tool packages supported by the new version 1.1 and the work which was done as part of HOPSA for each of the packages, as well as the implemented integration between the tool packages.

## 3.1 Supported Tool Packages

Table 1 gives an overview of all tool packages supported by the latest 1.1 version of the UNITE package and UNITE installer. Besides adding configuration, building, and installation support for Score-P components (scorep, opari2, otf2, cube), the new Scalasca Version 2 based on Score-P, the TAU tool from the University of Oregon, the BSC tools Extrae and Paraver (which in turn need support for the additional development packages libdwarf, libunwind, boost, wxpropgrid), and the HOPSA Light-weight monitoring module (LWM2), a significant amount of time was spent on testing of new tool versions once they were published to ensure the implemented UNITE support is still working.

***Table 1: Tool packages supported by the UNITE installer version 1.1***

| Package | Download site | OSV[1] | Work done as part of HOPSA |
|---|---|---|---|
| UNITE Installer | http://apps.fz-juelich.de/unite/ | — | • Enhanced to support additional packages (see below)<br>• More integration between tool and utility packages<br>• Support for more MPI library variants |
| UNITE Module Files | http://apps.fz-juelich.de/unite/ | 1.0 | • Enhanced to support additional packages (see below)<br>• Re-designed and re-implemented to allow update of existing UNITE installations without losing site-specific adaptations and changes |
| cube | http://www.scalasca.org/ | 3.3 | • Tested versions 3.4.x |
| cube4 | http://www.score-p.org/ | 4.0 | • Added in UNITE 1.1<br>• Tested versions 4.0.x, 4.1.x |
| Extrae(**) | http://www.bsc.es/paraver | 2.2 | • Added in UNITE 1.1<br>• Tested version 2.2.x, 2.3.x |
| Marmot | http://www.hlrs.de/organization/av/amt/research/marmot | 2.4.0 | — |
| LWM2 | http://www.hopsa-project.eu/tools/ | 1.1 | • Added in UNITE 1.1<br>• Tested versions 1.0, 1.1 |

[1] **O**ldest **S**upported **V**ersion

  

| | | | |
|---|---|---|---|
| OTF | http://www.tu-dresden.de/zih/otf/ | 1.5.0 | • Tested versions 1.7.x to 1.12.x |
| OTF2 | http://www.score-p.org/ | 1.0 | • Added in UNITE 1.1<br>• Tested versions 1.0, 1.1.x |
| opari2 | http://www.score-p.org/ | 1.0 | • Added in UNITE 1.1<br>• Tested versions 1.0.x |
| Paraver (**) | http://www.bsc.es/paraver | 4.3 | • Added in UNITE 1.1<br>• Tested versions 4.3.x to 4.4.x |
| pdtoolkit | http://www.cs.uoregon.edu/research/pdt/ | 3.15 | • Tested versions 3.16.x to 3.19.x |
| Scalasca-1.X | http://www.scalasca.org/ | 1.3.0 | • Tested versions 1.4.x |
| Scalasca-2.X | http://www.scalasca.org/ | 2.0 | • Added in UNITE 1.1 |
| Score-P | http://www.score-p.org/ | 1.0 | • Added in UNITE 1.1<br>• Tested version1.0, 1.1.x |
| TAU | http://tau.uoregon.edu/ | 2.20 | • Added in UNITE 1.1<br>• Tested versions 2.20.x to 2.22.x |
| VampirTrace | http://www.tu-dresden.de/zih/vampirtrace/ | 5.8 | • Tested version 5.9.x to 5.14.x |
| UniMCI | http://www.tu-dresden.de/zih/unimci/ | 1.0.1 | — |
| Vampir (*) | http://www.vampir.eu/ | 5.0 | • Support for new binary installer<br>• Tested version 7.x, 8.x |
| VampirServer (*) | http://www.vampir.eu/ | 1.0 | • Support for new binary installer<br>• Tested versions 7.x, 8.x |

**(*)** The commercial tool packages for Vampir and VampirServer are also supported, but these always need to be downloaded separately. The corresponding installation packages which you get after purchasing the Vampir software can simply be added to the `packages` directory of the UNITE installer. In addition, individual license file(s) have to be copied to the location reported by the UNITE installer configure script. For Linux systems, one can also download a time-limited demo version of Vampir 8 from the Website www.vampir.eu/download and use this package in the installation. The necessary demo license file will be emailed to you after giving your registration data at the Vampir website. Demo versions for other platforms or for VampirServer are available on request. For this, please contact service@vampir.eu.

**(**)** Extrae and Paraver need some additional utility packages for installation: boost, libunwind, and wxpropgrid (which are also included in the UNITE package for your convenience).

## 3.2 Implemented Tool Integration

The UNITE installer does not only allow configuring, building, and installing all the tool packages in a coherent and portable single installation, but also tries to configure and build the single tool packages in a way that there is maximum reuse of tool components and all possible integration between tools is enabled (see for example Figure 2).

| Package | Package Integration (and Necessary Configuration Options) |
|---|---|
| cube | • Configure and build remote control of Vampir (automatically configured if DBUS available) and Paraver components if available<br>`--with-paraver-cfg=<PARAVER_DIR>/<STATE-AS-IS-CFG-FILE>` |

  

| | |
|---|---|
| cube4 | • Configure and build remote control of Vampir (automatically configured if DBUS available) and Paraver components if available<br>`--with-paraver-cfg=<PARAVER_DIR>/<STATE-AS-IS-CFG-FILE>` |
| lwm2 | • Use external PAPI library if available<br>`--with-papi-header=<PAPI_DIR>/include`<br>`--with-papi-lib=<PAPI_DIR>/{lib,lib64}` |
| Scalasca-1.X | • Use external PAPI and OTF libraries and PDToolkit and Opari2 components if available<br>`--with-papi=<PAPI_DIR>`<br>`--with-pdt=<PDT_DIR>`<br>`--with-otf=<OTF_DIR>`<br>`--with-opari2=<OPARI2_DIR>` |
| Scalasca-2.X | • Use external OTF2 library and CUBE4 component if available<br>`--with-otf2=<OTF2_DIR>`<br>`--with-cube=<CUBE4_DIR>/bin` |
| Marmot | • Use external CUBE component if available<br>`-DMARMOT_USE_CUBE=ON`<br>`-DCUBE_CONFIG=<CUBE_DIR>/bin/cube-config` |
| UniMCI | • Configure for MARMOT if available<br>`-DMARMOT_HOME=<MRMT_DIR>`<br>`-DUSED_MPI_CHECKER=MARMOT` |
| VampirTrace | • Use external PAPI and UNIMCI libraries and PDToolkit component if available<br>`--with-papi-dir=<PAPI_DIR>`<br>`--with-unimci-config=<MRMT_DIR>/bin/unimci-config` |
| Extrae | • Use external PAPI library if available<br>`--with-papi=<PAPI_DIR>`<br>• Use separately configured and installed boost, libdwarf, and libunwind development packages |
| Paraver | • Use separately configured and installed wxpropgrid development package |
| Score-P | • Use external PAPI and OTF2 libraries and PDToolkit, Opari2 and CUBE4 components if available<br>`--with-papi-header=<PAPI_DIR>/include`<br>`--with-papi-lib=<PAPI_DIR>/{lib,lib64}`<br>`--with-pdt=<PDT_DIR></ARCH>/bin`<br>`--with-otf2=<OTF2_DIR>`<br>`--with-opari2=<OPARI2_DIR>/bin`<br>`--with-cube=<CUBE4_DIR>/bin` |
| TAU | • Configure and build MPI, OpenMP, MPI/OpenMP versions each to<br>  • Use external PAPI and OTF libraries and PDToolkit component if available<br>`-papi=<PAPI_DIR>`<br>`-pdt=<PDT_DIR>`<br>`-otf=<OTF_DIR>`<br>  • Use TAU internal, Scalasca, VampirTrace, or Score-P measurement system<br>`-scalasca=<SCALASCA_DIR>`<br>`-vampirtrace=<VT_DIR>`<br>`-scorep=<SCOREP_DIR>` |

  

# 4. Conclusion

**UNITE** (**UN**iform **I**ntegrated **T**ool **E**nvironment) provides a robust, portable, and integrated environment for the debugging and performance analysis of parallel MPI, OpenMP, and hybrid MPI/OpenMP programs on high-performance compute clusters. It consists of a set of well-accepted portable, mostly open-source tools.

As high-performance clusters often provide multiple MPI libraries and compiler suites for parallel programming, which requires separate versions of tools for each MPI library/compiler combination, and HPC centres have different policies on how and where to install different software packages as well as how to name the different versions, UNITE defines a portable and coherent tool installation setup by

- specifying exactly how and where to install the different versions of tool software packages (including integrating the tools to the maximum possible degree),
- defining standard module names for tools and their different versions, and
- supplying pre-defined module files which provide standardized, well-tested tool configurations,
- **but** still being flexible enough to be able to co-exist with site-local installations, restrictions, and policies.

This way**,** HPC application developers only need to learn once how to invoke and facilitate the debugging and performance tools for their applications (at least at sites which use the UNITE package).

UNITE helps HPC system administrators by providing a ”meta”-installation tool (the UNITE installer) which is capable of configuring, building, and installing all supported tools as a common package according to the UNITE specifications but hiding tool-specific aspects of the various phases. It can also be used to update an existing UNITE installation with new tools or tool versions.

On the UNITE website [14], a single package consisting of the UNITE installer, the UNITE module files, as well as a large set of open-source debugging and performance tools together with a user guide (for application developers) and an installation guide (for system administrators) is provided.

  

# 5. Bibliography

[1] M. Geimer, F. Wolf, B. J. N. Wylie, E. Abraham, D. Becker, and B. Mohr. The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, 22(6):702–719, April 2010.

[2] M. Geimer, F. Wolf, B. J. N. Wylie, B. Mohr: A scalable tool architecture for diagnosing wait states in massively parallel applications. *Parallel Computing*, 35(7):375-388, July 2009.

[3] B. J. N. Wylie, M. Geimer, F. Wolf: Performance measurement and analysis of large-scale parallel applications on leadership computing systems. *Scientific Programming*, 16(2-3):167-181, 2008, Special Issue Large-Scale Programming Tools and Environments.

[4] E. Hagersten, M. Nilsson and M. Vesterlund, Improving Cache Utilization Using Acumem VPE, *Tools for High-Performance Computing 2008*, III, 115-135, DOI: 10.1007/978-3-540-68564-7_8

[5] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. Müller and W.E. Nagel, "The Vampir Performance Analysis Tool-Set", *Tools for High Performance Computing*, pp 139-155, Springer Verlag, 2008.

[6] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorf, K. Diethelm, D. Eschweiler, M. Gerndt, D. Lorenz, A. D. Malony, W. E. Nagel, Y. Oleynik, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, F. Wolf: *Score-P - A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir,* Proceedings of 5th Parallel Tools Workshop, 2011.

[7] J. Labarta, Trace-Based Tools, *Performance Tuning of Scientific Applications*, Edited by D. H. Bailey, R. F. Lucas and S. W. Williams, pp. 87–122, 2011.

[8] A. Snavely, X. Gao, C. Lee, L. Carrington, N. Wolter, J. Labarta, J. Giménez, P. Jones, *Performance Modeling of HPC Applications*, Proceedings ParCo 2003.

[9] V. Pillet et al.: PARAVER: A Tool to Visualize and Analyze Parallel Code, in: 18th World OCCAM and Transputer User Group Technical Meeting, April 1995.

[10] T-Platforms, Moscow, Russia. Clustrx HPC Software. http://www.t-platforms.com/products/software/clustrxproductfamily.html, last accessed September 2012.

[11] A.V. Adinets, P.A. Bryzgalov, V. Vad, Voevodin, S.A. Zhumatiy, D.A. Nikitenko. *About an approach to monitoring, analysis and visualization of jobs on cluster system* (In Russian). In: Numerical Methods and Programming, vol. 12, pp. 90–93, 2011.

[12] Bernd Mohr, Vladimir Voevodin, Judit Giménez, Erik Hagersten, Andreas Knüpfer, Dmitry A. Nikitenko, Mats Nilsson, Harald Servat, Aamer Shah, Frank Winkler, Felix Wolf, and Ilya Zhukov: *The HOPSA Workflow and Tools.* In: Proceedings of the 6th International Parallel Tools Workshop, Stuttgart, September 2012, Springer. To appear.

[13] The module software, http://modules.sourceforge.net/

[14] UNITE website, http://apps.fz-juelich.de/unite/

[15] EU Itea2 project ParMA, http://www.parma-itea2.org/

  

# 6. Annexes

## 6.1 Annex A: UNITE User Guide

## 6.2 Annex B: UNITE Installation Guide